NUXT 3 ESSENTIALS CHEAT SHEET

Insidewebdev

STARTING A NEW PROJECT

Create a project using nuxi:

```
$ npx nuxi init <project-name>
$ cd <project-name>
$ npm install        Installs dependencies
$ npm run dev        Runs the project
```

FOLDER STRUCTURE

- **ASSETS -** Uncompiled assets (like Sass, fonts Images)
- **COMPONENTS -** Components are automatically imported based on the folder and file name
- **COMPOSABLES -** Directory to automatically import your composables into your application
- **CONTENT -** Contains files to create a file based CMS
- **LAYOUTS -** Application layouts
- **MIDDLEWARE -** Custom functions to run before each page
- **PAGES -** Application views & routes from which the router is dynamically generated
- **PLUGINS -** JS plugins run before Vue.js init
- **SERVER -** Create serverless functions

STATE MANAGEMENT

Nuxt provides `useState` composable to create a reactive and SSR-friendly shared state across components.

🚨 Never define **const state = ref()** outside of **<script setup>** or **setup()** function.

✅ Instead use **const useX = () => useState('x')**

Basic

```
<script setup>
const counter = useState('counter', () =>  Math.random() * 1000)
</script>
```

Shared state

Composables defined inside ***~/composables*** folder are auto-imported and with that we can share states and import them across the app.

composables/states.ts

```
export const useCounter = () => useState<number>('counter', () => 0)
export const useColor = () => useState<string>('color', () => 'pink')
```

app.vue

```
<script setup>
const color = useColor() // Same as useState('color')
</script>

<template>
  <p>Current color: {{ color }}</p>
</template>
```

PAGES

Nuxt generates each app route from the folder `pages` directory.

```
pages
  index.vue              loads the root path /
  posts
    [...slug].vue        [...x] catch all route /posts/my-slug
    index.vue            /posts
  users-[group]          we can also add params inside a folder . /users-customer
    [id].vue             [ ] defines a dynamic route with a param. /users-admin/123
```

Access directly inside the template

pages/users-[group]/[id].vue

```
<template>
  <p>{{ $route.params.group }} - {{ $route.params.id }}</p>
</template>
```

Access inside script tag using Composition API

pages/users-[group]/[id].vue

```
<script setup>
const route = useRoute()

const { group, id } = route.params

</script>
```

DATA FETCHING

Nuxt provides **useFetch**, **useLazyFetch**, **useAsyncData** and **useLazyAsyncData** to handle data fetching within your application.

⚠️ **useFetch**, **useLazyFetch**, **useAsyncData** and **useLazyAsyncData** only work during setup or Lifecycle Hooks

useFetch()

```
<script setup>
const { data: count, pending, refresh, error  } = await useFetch('/api/count')
</script>

<template>
  Page visits: {{ count }}
</template>
```

useAsyncData()

```
<script setup>
const { data } = await useAsyncData('count', () => $fetch('/api/count'))
</script>

<template>
  Page visits: {{ data }}
</template>
```

👉 Difference between useFetch and useAsyncData:
useFetch receives a URL and gets that data, whereas **useAsyncData** might have more complex logic.
useFetch(url) is nearly equivalent to ***useAsyncData(url, () => $fetch(url))***

useLazyFetch() and useLazyAsyncData()

These composables behave identically to ***useFetch*** and ***useAsyncData*** with the ***lazy: true*** option set. In other words, the async function does not block navigation.

NUXT 3 ESSENTIALS CHEAT SHEET

Insidewebdev

SERVER ROUTES

Files inside the ***~/server/api*** are automatically prefixed with ***/api*** in their route. For adding server routes without ***/api*** prefix, you can instead put them into ***~/server/routes*** directory.

server/api/route.post.js

```
import { sendError } from "h3"

export default defineEventHandler(async (event) => {
    const config = useRuntimeConfig()  // Get the runtime config

    const query = useQuery(event) // Get the query from the event
    const body = await useBody(event) // Get the body from the event
    const headers = useHead(event) // Get the headers from the event
    const cookies = useCookies(event) // Get the cookies from the event

    // Throw error
    if(something) {
        return sendError(event, createError({
            statusCode: 400,
            statusMessage: ‘Error message’
        }))
    }

    return {
        // returns a response object
    }
})
```

Matching route params

Server routes can use dynamic parameters within brackets in the file name like ***/api/hello/[name].ts*** and accessed via ***event.context.params***.
Catching all routes its as easy as using the spread operator ***[...name]***

/server/api/hello/[name].ts

```
export default defineEventHandler(event => `Hello, ${event.context.params.name}!`)
```

Matching HTTP Method

Handle file names can be suffixed with ***.get***, ***.post***, ***.put***, ***.delete***, ... to match request’s.

/server/api/test.**get**.ts

```
export default defineEventHandler(() => 'Test get handler')
```

/server/api/test.**post**.ts

```
export default defineEventHandler(() => 'Test post handler')
```

SERVER MIDDLEWARE

Nuxt will automatically read in any file in the ***~/server/middleware*** to create server middleware for your project.

server/middleware/auth.js

```
export default defineEventHandler((event) => {
  console.log('New request: ' + event.req.url)
})
```

TELEPORT

```
<template>
  <button @click="open = true">
    Open Modal
  </button>
  <Teleport to="body">
    <div v-if="open" class="modal">
      <p>Hello from the modal!</p>
      <button @click="open = false">
        Close
      </button>
    </div>
  </Teleport>
</template>
```

RUNTIME CONFIG

Expose runtime config

nuxt.config.ts

```
export default defineNuxtConfig({
  runtimeConfig: {
    apiSecret: process.env.API_SECRET,
    public: {
      apiBase: process.env.API_BASE,
    }
  },
})
```

Accessing runtime config

Client

```
<script setup>
const config = useRuntimeConfig()
console.log('Runtime config:', config)
if (process.server) {
  console.log('API secret:',
config.apiSecret)
}
</script>
```

Server

```
export default defineEventHandler(async
(event) => {
    const config = useRuntimeConfig()

    return {
        // returns a response object
    }
})
```

For more quality programming courses

insidewebdev